ELSEVIER Journal of Computational and Applied Mathematics 74 (1996) 71–89

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Some large-scale matrix computation problems

Zhaojun Bai[a, 1], Mark Fahey[a, 1], Gene Golub[b, *, 2]

[a]*Department of Mathematics, University of Kentucky, Lexington, KY 40506, USA*
[b]*Scientific Computing and Computational Mathematics Program, Computer Science Department, Stanford University, Stanford, CA 94305, USA*

Abstract

There are numerous applications in physics, statistics and electrical circuit simulation where it is required to bound entries and the trace of the inverse and the determinant of a large sparse matrix. All these computational tasks are related to the central mathematical problem studied in this paper, namely, bounding the bilinear form $\boldsymbol{u}^{\mathrm{T}} f(A)\boldsymbol{v}$ for a given matrix A and vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, where f is a given smooth function and is defined on the spectrum of A. We will study a practical numerical algorithm for bounding the bilinear form, where the matrix A is only referenced through matrix-vector multiplications. A Monte Carlo method is also presented to efficiently estimate the trace of the inverse and the determinant of a large sparse matrix.

Keywords: Bilinear form; Gaussian quadrature; Trace; Determinant; Matrix inverse; Monte Carlo simulation; Probabilistic bound

AMS classification: 65F10

1. Introduction

The central problem studied in this paper is to estimate a lower bound L and/or an upper bound U, such that

$$L \leqslant \boldsymbol{u}^{\mathrm{T}} f(A)\, \boldsymbol{v} \leqslant U, \tag{1}$$

where A is a $n \times n$ given real matrix, $\boldsymbol{u}$ and $\boldsymbol{v}$ are given n-vectors, and f is a given smooth function and is defined on the spectrum of the matrix A. For example, if A is an $n \times n$ nonsingular matrix,

* Corresponding author.

[1] The authors were supported in part by an NSF grant ASC-9313958 and in part by an DOE grant DE-FG03-94ER25219 via subcontracts from University of California at Berkeley. Fahey is also supported by a fellowship from Center for Computational Sciences, University of Kentucky.

[2] The work of this author was in part supported by NSF under grant CCR-9505393.

$f(\lambda) = 1/\lambda$, and $\boldsymbol{u} = \boldsymbol{v} = \boldsymbol{e}_i$, the ith column of the identity matrix I, then $\boldsymbol{u}^{\mathrm{T}}f(A)\boldsymbol{v} = (A^{-1})_{ii}$. The problem (1) is to bound the ith diagonal element of the inverse of the matrix A. If we take the sum of all diagonal elements of the inverse of the matrix A, then a related problem is to estimate bounds on the trace of the inverse of A, $\mathrm{tr}(A^{-1})$. Later, we will see that estimating bounds of the determinant $\det(A)$ of A is also a related problem.

When the matrix size n is small, say $n \leqslant 200$, we can compute the quantity $\boldsymbol{u}^{\mathrm{T}}f(A)\boldsymbol{v}$ explicitly by using dense matrix computation methods [8]. In general, such methods may require $\mathcal{O}(n^2)$ memory storage and $\mathcal{O}(n^3)$ floating point operations. Therefore, when n is large, it is impractical to compute $\boldsymbol{u}^{\mathrm{T}} f(A)\boldsymbol{v}$ explicitly. In this paper, we will study numerical methods in which the matrix in question is only referenced in the form of matrix–vector products. Because of this feature, the methods are well-suited for large sparse matrices or large structured dense matrices for which matrix–vector products can be computed cheaply. They will normally take about $\mathcal{O}(3n + \kappa)$ words of memory and $\mathcal{O}(j\nu)$ floating point operations, where κ is the required memory for storing the matrix and/or forming matrix–vector products, ν is the cost of matrix–vector product and j is the number of iterations.

Using variational principles, Robinson and Wathen have studied a special case of the problem (1), namely, bounding the entries of the inverse of a matrix [18]. Golub, Meurant and Strakos have studied the problem (1) when the matrix A is symmetric positive definite [6, 7]. They closely examined the application of their approach for bounding the entries of the inverse of a matrix. Quadrature rules, orthogonal polynomial theory and the underlying Lanczos procedure are the tools used in their approach.

In this paper, we will follow the work in [6, 7] and further develop it in the following aspects. First, we will discuss some practical implementation issues of algorithms. Second, we will show how to use the proposed algorithms for bounding the quantities $(A^{-1})_{ij}$, $\mathrm{tr}(A^{-1})$ and $\det(A)$, where the matrix A is not necessarily symmetric positive definite. Third, we will present a Monte Carlo approach for estimating the quantities $\mathrm{tr}(A^{-1})$ and $\det(A)$, which significantly reduces the computational cost and obtain a truly practical method for dealing with large-scale matrices.

A number of applications of problem (1) have been discussed in [7], such as estimating the accuracy of the CG method for solving large linear systems of equations and solving constraint quadratic optimization. New sources of applications are in the fields of fractals [19, 12, 22] and lattice Quantum Chromodynamics (QCD) [10, 20, 2]. In fact, the QCD applications are the main motivation of our current studies. It is said that a large fraction of the world's supercomputer time is being consumed by physicists in lattice QCD to meet their stringent numerical computation demands. Some of their computational kernels are focused on solving large scale matrix computation problems, such as computing the trace of the inverse and the determinant of matrices of order millions. Numerical analysts have been advised for years that these quantities are too numerically sensitive to compute. Computational physicists, with little or no help from numerical analysts, have developed numerical techniques to compute these quantities. In this paper, we aim to develop practical numerical techniques to tackle these difficult matrix computation problems that have eluded us for years. We believe that our approaches are more efficient than those currently being used by practitioners.

The rest of this paper is organized as follows: In Section 2, we review the basic idea of algorithms presented in [6, 7], and discuss a number of practical implementation issues of algorithms. Section 3 discusses the applications of the approaches for estimating the bounds of the entries and trace of the inverse of a matrix and the determinant. A Monte Carlo approach and the related confidence bounds are studied in Section 4. Section 5 collects some numerical results. This paper is concluded with open problems and future work.

2. Basic algorithm

In this section, we first review the approach presented in [6, 7] for estimating the quantity $\boldsymbol{u}^{\mathrm{T}}f(A)\boldsymbol{v}$ and then discuss some implementation issues. We will assume that the matrix A is symmetric positive definite. In the next section, we will show how to use these algorithms to bound the entries of the inverse of a matrix A, $\operatorname{tr}(A^{-1})$ and $\det(A)$, where A is not necessarily symmetric positive definite.

2.1. Main idea

The main idea of the approaches proposed in [6, 7] is to first transform the central problem (1) to a Riemann–Stieltjes integral problem, and then use quadrature rules to approximate the integral, which brings the orthogonal polynomial theory and the underlying Lanczos procedure into the picture. Let us go through the main idea. Since A is symmetric, the eigen-decomposition of A is given by $A = Q^{\mathrm{T}}\Lambda Q$, where Q is an orthogonal matrix and Λ is a diagonal matrix with increasingly ordered diagonal elements λ_i. Then

$$\boldsymbol{u}^{\mathrm{T}}f(A)\boldsymbol{v} = \boldsymbol{u}^{\mathrm{T}}Q^{\mathrm{T}}f(\Lambda)Q\boldsymbol{v} = \tilde{\boldsymbol{u}}^{\mathrm{T}}f(\Lambda)\tilde{\boldsymbol{v}} = \sum_{i=1}^{n} f(\lambda_i)\tilde{u}_i\tilde{v}_i,$$

where $\tilde{\boldsymbol{u}} = (\tilde{u}_i) \equiv Q\boldsymbol{u}$ and $\tilde{\boldsymbol{v}} = (\tilde{v}_i) \equiv Q\boldsymbol{v}$. The last sum can be considered as a Riemann–Stieltjes integral

$$I[f] = \boldsymbol{u}^{\mathrm{T}}f(A)\boldsymbol{v} = \int_a^b f(\lambda)\,\mathrm{d}\mu(\lambda),$$

where the measure $\mu(\lambda)$ is a piecewise constant function defined by

$$\mu(\lambda) = \begin{cases} 0 & \text{if } \lambda < a = \lambda_1, \\ \sum_{j=1}^{i} \tilde{u}_j\tilde{v}_j & \text{if } \lambda_i \leqslant \lambda < \lambda_{i+1}, \\ \sum_{j=1}^{n} \tilde{u}_j\tilde{v}_j & \text{if } b = \lambda_n \leqslant \lambda. \end{cases}$$

A way to obtain an estimate for the Riemann–Stieltjes integral is to use Gauss-type quadrature rules [3, 1]. The general quadrature formula is of the form

$$I[f] = \sum_{j=1}^{N} \omega_j f(\theta_j) + \sum_{k=1}^{M} v_k f(\tau_k), \tag{2}$$

where the weights $\{\omega_j\}$ and $\{v_k\}$ and the nodes $\{\theta_j\}$ are unknown and to be determined. The nodes $\{\tau_k\}$ are prescribed. The integration error (remainder) is

$$R[f] = \int_a^b f(\lambda)\,\mathrm{d}\mu(\lambda) - I[f].$$

If $M = 0$, then it is the well-known Gauss rule. If $M = 1$ and $\tau_1 = a$ or $\tau_1 = b$, it is the Gauss–Radau rule. If $M = 2$ and $\tau_1 = a$ and $\tau_2 = b$, it is the Gauss–Lobatto rule. The sign of $R[f]$ determines whether the quadrature formula $I[f]$ is a lower bound (if $R[f] > 0$) or an upper lower bound (if $R[f] < 0$) of the quantity $\boldsymbol{u}^{\mathrm{T}} f(A)\boldsymbol{v}$.

We will focus on the case $\boldsymbol{u} = \boldsymbol{v}$ in the following discussion. In this case, the measure $\mu(\lambda)$ is a positive increasing piecewise constant function. We note that the case $\boldsymbol{u} \neq \boldsymbol{v}$ can be reduced to the case $\boldsymbol{u} = \boldsymbol{v}$ by the polarization expression

$$\boldsymbol{u}^{\mathrm{T}} f(A)\boldsymbol{v} = \tfrac{1}{4}(\boldsymbol{y}^{\mathrm{T}} f(A)\boldsymbol{y} - \boldsymbol{z}^{\mathrm{T}} f(A)\boldsymbol{z}), \tag{3}$$

where $\boldsymbol{y} = \boldsymbol{u} + \boldsymbol{v}$ and $\boldsymbol{z} = \boldsymbol{u} - \boldsymbol{v}$.

2.2. Basic algorithm

Let us briefly recall how the weights and the nodes in the quadrature formula are obtained. First, we know that a sequence of polynomials $\{p_i(\lambda)\}_{i=0}^{\infty}$ can be defined such that they are orthonormal with respect to $\mu(\lambda)$, i.e.,

$$\int_a^b p_i(\lambda)p_j(\lambda)\,\mathrm{d}\mu(\lambda) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases}$$

where it is assumed that $\int \mathrm{d}\mu = 1$. The sequence of orthonormal polynomials satisfies a three-term recurrence

$$\gamma_j p_j(\lambda) = (\lambda - \alpha_j)p_{j-1}(\lambda) - \gamma_{j-1}p_{j-2}(\lambda), \tag{4}$$

for $j = 1, 2, \ldots$ with $p_{-1}(\lambda) \equiv 0$ and $p_0(\lambda) \equiv 1$. Writing the recurrence in matrix form, we have

$$\lambda \boldsymbol{p}(\lambda) = T_j \boldsymbol{p}(\lambda) + \gamma_j p_j(\lambda)\boldsymbol{e}_j,$$

where

$$\boldsymbol{p}(\lambda)^{\mathrm{T}} = [p_0(\lambda), p_1(\lambda), \ldots, p_{j-1}(\lambda)], \quad \boldsymbol{e}_j^{\mathrm{T}} = [0, 0, \ldots, 1],$$

and

$$T_j = \begin{pmatrix} \alpha_1 & \gamma_1 & & & \\ \gamma_1 & \alpha_2 & \gamma_2 & & \\ & \gamma_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \gamma_{j-1} \\ & & & \gamma_{j-1} & \alpha_j \end{pmatrix}.$$

Then in the Gauss quadrature rule, the eigenvalues of T_j (which are the zeros of $p_j(\lambda)$) are the nodes θ_j. The weights ω_j are the squares of the first elements of the normalized (i.e., unit norm) eigenvectors of T_j [1].

In the Gauss–Radau and Gauss–Lobatto rules, the nodes $\{\theta_j\}$, $\{\tau_k\}$ and weights $\{\omega_j\}$, $\{v_j\}$ come from eigenvalues and the squares of the first elements of the normalized eigenvectors of an adjusted tridiagonal matrix of $\tilde{T}_{j'}$, which has the prescribed eigenvalues a and/or b, i.e., a and/or b are roots of the polynomial $p_{j+1}(\lambda)$ [6].

To this end, we recall that the classical Lanczos procedure is an elegant way to compute the orthonormal polynomials $p_j(\lambda)$ [14, 6]. We have the following algorithm in summary form.

Algorithm 1. Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\lambda(A) \subset [a,b]$, $\boldsymbol{u} \in \mathbb{R}^n$ and f is a smooth function in the interval $[a,b]$. Then the following algorithm computes a lower bound L and/or an upper bound U of the quantity $\boldsymbol{u}^{\mathrm{T}} f(A)\boldsymbol{u}$ by using the Gauss, Gauss–Radau or Gauss–Lobatto rules.

```
• Let $x_0 = u/\|u\|_2$, and $x_{-1} = 0$ and $\gamma_0 = 0$
• For $j = 1, 2, \ldots$, until convergence
    1. $\alpha_j = x_{j-1}^{\mathrm{T}} A x_{j-1}$
    2. $r_j = Ax_{j-1} - \alpha_j x_{j-1} - \gamma_{j-1} x_{j-2}$
    3. $\gamma_j = \|r_j\|_2$
    4. For Gauss–Radau or Gauss–Lobatto rules, adjust $T_j$ to have the prescribed eigenvalue (s)
       $a$ and/or $b$, denote the resulting matrix $\tilde{T}_{j'}$ (see Remark 3 below)
    5. Compute eigenvalues $\theta_k$ and the first elements $\omega_k$ of the eigenvectors of $\tilde{T}_{j'}$
    6. Compute $I_j = \sum_{k=1}^{j'} \omega_k^2 f(\theta_k)$
    7. If using the Gauss–Radau rule, repeat steps 5–7 once
    8. Check for convergence (see Remark 4 below), continue if necessary
    9. $x_j = r_j/\gamma_j$
• End
• $L = \|u\|_2^2 I_j$ and/or $U = \|u\|_2^2 I_j$
```

The following remarks are in order:

Remark 1. For the Gauss–Radau and Gauss–Lobatto rules, we need to find an interval $[a,b]$ such that $\lambda(A) \subset [a,b]$. A simple way to find a and b is to use Gerschgorin circles. If the lower bound computed by the Gerschgorin circles is less than or equal to 0, we simply set a to be a small number (say 10^{-4}).

Other sophisticated algorithms exist for estimating a and b. For example, we can use the Lanczos method for estimating the extreme eigenvalues [8].

Remark 2. On the return of the algorithm, for the Gauss rule, from the expression of $R[f]$, we know that if $f^{(2n)}(\xi) > 0$ for any n and ξ, $a < \xi < b$, then I_j is a lower bound L.

For the Gauss–Radau rule, if $f^{(2n+1)}(\xi) < 0$ for any n and ξ, $a < \xi < b$, then I_j (with b as a prescribed eigenvalue of $\tilde{T}_{j'}$) is a lower bound L, and I_j (with a as a prescribed eigenvalue of $\tilde{T}_{j'}$) is an upper bound U.

For the Gauss–Lobatto rule, if $f^{(2n)}(\xi) > 0$ for any n and ξ, $a < \xi < b$, then I_j is an upper bound U.

Remark 3. If the Gauss rule is used, no adjustment of the tridiagonal matrix T_j is necessary, i.e., $\tilde{T}_{j'} = T_j$.

If the Gauss–Radau rule is used, steps 5–7 are executed twice. We need to construct the polynomial $p_{j+1}(\lambda)$ such that $p_{j+1}(a) = 0$ and $p_{j+1}(b) = 0$, respectively, i.e., we need to extend the matrix T_j to

$$\tilde{T}_{j'} = \begin{bmatrix} T_j & \gamma_j e_j \\ \gamma_j e_j^{\mathrm{T}} & \phi \end{bmatrix},$$

where the parameter ϕ is chosen such that a and b is an eigenvalue of $\tilde{T}_{j'}$, respectively. From [4, 6], we know that $\phi = a + \delta_j$ and $\phi = b + \delta_j$, respectively, where δ_j is the last component of the solution $\boldsymbol{\delta}$ of the tridiagonal system

$$(T_j - aI)\boldsymbol{\delta} = \gamma_j^2 \boldsymbol{e}_j \quad \text{or} \quad (T_j - bI)\boldsymbol{\delta} = \gamma_j^2 e_j.$$

If the Gauss–Lobatto rule is used, we need to construct $p_{j+1}(\lambda)$ such that $p_{j+1}(a) = p_{j+1}(b) = 0$, i.e., T_j is updated to

$$\tilde{T}_{j'} = \begin{bmatrix} T_j & \psi e_j \\ \psi e_j^{\mathrm{T}} & \phi \end{bmatrix}$$

where the parameters ϕ and ψ are chosen so that a and b are eigenvalues of $\tilde{T}_{j'}$. Again, from [4, 6], we know that

$$\phi = \frac{\delta_j b - \mu_j a}{\delta_j - \mu_j} \quad \text{and} \quad \psi^2 = \frac{b - a}{\delta_j - \mu_j},$$

where δ_j and μ_j are the last components of the solutions $\boldsymbol{\delta}$ and $\boldsymbol{\mu}$ of the tridiagonal systems

$$(T_j - aI)\boldsymbol{\delta} = e_j \quad \text{and} \quad (T_j - bI)\boldsymbol{\mu} = e_j.$$

Remark 4. As for any iterative method, we first should have a parameter *maxit* to limit the maximum number of the iterations (i.e., $j < maxit$). The following stopping criterion may be used:

$$|I_j - I_{j-1}| \leqslant \varepsilon |I_j|,$$

where ε is a user specified tolerance value. This criterion tells that for the approximation I_j to the lower bound L or the upper bound U, we have

$$|L - I_j| \leqslant |L - I_{j-1}| + \varepsilon |I_j| \quad \text{or} \quad |U - I_j| \leqslant |U - I_{j-1}| + \varepsilon |I_j|.$$

Therefore, the iteration stops if the error is no longer decreasing or decreasing too slowly.

Remark 5. The combination of the steps 1–4 and 10 is a step of the standard symmetric Lanczos procedure. Note that the matrix A in question is only referenced here in the form of the matrix–vector product. The Lanczos procedure can be implemented with only $3n$-vectors in the fast memory [8]. This is the major storage requirement for the algorithm and is an attractive feature for large-scale problems.

Remark 6. As pointed out in [6, 7], we need not always explicitly compute the eigenvalues and eigenvectors of the tridiagonal matrix $\tilde{T}_{j'}$. It can be shown that

$$\sum_{k=1}^{j'} \omega_k^2 f(\theta_k) = e_1^T f(\tilde{T}_{j'}) e_1. \tag{5}$$

Therefore, in the case where the (1, 1) entry of $f(\tilde{T}_{j'})$ can easily be computed, the computation of the eigenvalues and eigenvectors of $\tilde{T}_{j'}$ is not necessary (see Section 3).

3. Bounds on the quantities $(A^{-1})_{ij}$, $\operatorname{tr}(A^{-1})$ and det (A)

In this section, we show how to use Algorithm 1 (presented in Section 2) by bounding the quantities $(A^{-1})_{ij}$, $\operatorname{tr}(A^{-1})$ and $\det(A)$. We first discuss the case where the matrix A is symmetric positive definite and then extend to the case where the matrix A is not symmetric positive definite.

3.1. Matrix A is symmetric positive definite

3.1.1. Bounds on $(A^{-1})_{ii}$

To bound a diagonal element of the inverse of A, $(A^{-1})_{ii}$, for a given i, $1 \leqslant i \leqslant n$, we let $\boldsymbol{u} = \boldsymbol{v} = \boldsymbol{e}_i$ and $f(\lambda) = 1/\lambda$, $0 < a < \lambda < b$, then $\boldsymbol{u}^T f(A) \boldsymbol{u} = (A^{-1})_{ii}$. Note that $f^{(2n+1)}(\lambda) = -(2n+1)!\,\lambda^{-(2n+2)}$ and $f^{(2n)}(\lambda) = (2n)!\,\lambda^{-(2n+1)}$. By Algorithm 1 with the Gauss rule, at convergence, I_j gives a lower bound L of $(A^{-1})_{ii}$. With the Gauss–Radau rule, we obtain both lower bound L and upper bound U of the diagonal entry $(A^{-1})_{ii}$. The Gauss–Lobatto rule will give an upper bound U of $(A^{-1})_{ii}$.

In this case, we can avoid computing the eigenvalues and eigenvectors of the tridiagonal matrix $\tilde{T}_{j'}$ explicitly. From Eq. (5), we only need to compute the (1, 1) element of $(\tilde{T}_{j'})^{-1}$, which can be computed recursively. The formulation detail has been worked out in [6].

3.1.2. Bounds on $(A^{-1})_{ij}$

If $\boldsymbol{u} = \boldsymbol{e}_i$ and $\boldsymbol{v} = \boldsymbol{e}_j$ and $f(\lambda) = 1/\lambda$, $0 < a < \lambda < b$, then $\boldsymbol{u}^T f(A) \boldsymbol{v} = (A^{-1})_{ij}$. Using (3), we have

$$(A^{-1})_{ij} = \tfrac{1}{4}(\boldsymbol{y}^T A^{-1} \boldsymbol{y} - \boldsymbol{z}^T A^{-1} \boldsymbol{z}),$$

where $\boldsymbol{y} = \boldsymbol{e}_i + \boldsymbol{e}_j$ and $\boldsymbol{z} = \boldsymbol{e}_i - \boldsymbol{e}_j$. Thus, the bounds for off-diagonal entry $(A^{-1})_{ij}$ can also be obtained by using Algorithm 1.

3.1.3. Bounds on $tr(A^{-1})$

We know from above that a lower bound L_i and/or upper bound U_i can be estimated for each diagonal element $(A^{-1})_{ii}$, $i = 1, 2, \ldots, n$. Thus, a lower bound of $\operatorname{tr}(A^{-1})$ is given by $\sum_{i=1}^n L_i$ and an upper bound is given by $\sum_{i=1}^n U_i$. However, this would require executing Algorithm 1 n times. Although it only takes about $\mathcal{O}(jv)$ flops (the main cost is for the matrix–vector product) to run Algorithm 1 once, if we run the algorithm n times, it will cost $\mathcal{O}(njv)$ flops. It ends up with the same computational complexity as the dense matrix computation methods which is unacceptable in practice. In the next section, we will propose a Monte Carlo approach to estimate $\operatorname{tr}(A^{-1})$ which only requires executing Algorithm 1 m times, where m is significantly smaller than n, say $m = 30$ to 50. It results in a significant saving in terms of computational costs.

3.1.4. Bounds on det(A)

It can easily be verified by using the eigen-decomposition that for a symmetric positive-definite matrix A:

$$\ln(\det(A)) = \operatorname{tr}(\ln(A)), \tag{6}$$

i.e.,

$$\det(A) = \exp(\operatorname{tr}(\ln(A))). \tag{7}$$

Therefore, if we let $f(\lambda) = \ln \lambda$, then the problem of estimating $\det(A)$ is reduced to bound the trace of the matrix natural logarithm function $\ln(A)$, i.e., $\sum_{i=1}^{n} (\ln(A))_{ii}$. Note that $f^{(2n+1)}(\lambda) = (2n)!\lambda^{-(2n+1)} > 0$ and $f^{(2n)}(\lambda) = -(2n-1)!\lambda^{-(2n)} < 0$ for any n and $0 < a < \lambda < b$, then we know that by applying Algorithm 1 with the Gauss rule, at convergence, I_j gives an upper bound U_i of $(\ln(A))_{ii}$. A lower bound L_i and an upper bound U_i of $(\ln(A))_{ii}$ can be obtained by using Algorithm 1 with the Gauss–Radau rule. A lower bound of U_i can be obtained by using Algorithm 1 with the Gauss–Lobatto rule.

However, if we try to bound each $(\ln(A))_{ii}$ for $1 \leqslant i \leqslant n$, it requires running Algorithm 1 n times and could be too costly to use in practice. The Monte Carlo approach to be discussed in Section 4 will relieve this computational burden in practice.

3.2. Matrix A is not symmetric positive definite

In some applications, such as in lattice QCD, the matrix A involved may not be symmetric positive definite. It may not even be symmetric. In this section, we show how we can use the techniques developed in Section 2 to bound $(A^{-1})_{ij}$, $\operatorname{tr}(A^{-1})$, and $\det(A)$, where A is nonsymmetric positive definite. The main observation of such a transformation is rather simple. Note that if A is nonsingular, then $A^{\mathrm{T}}A$ is symmetric positive definite, and furthermore, from $(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}A = I$, we have

$$A^{-1} = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}. \tag{8}$$

Of course, in practice, we will never form the matrix $A^{\mathrm{T}}A$ explicitly. The matrix–vector product $A^{\mathrm{T}}Ax$ is computed in two matrix–vector products. An alternative approach is to apply Lanczos procedure to bidiagonalize A [8].

3.2.1. Bounds on $(A^{-1})_{ij}$

From (8), we have

$$e_i^{\mathrm{T}}A^{-1}e_j = e_i^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}e_j = e_i^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}\boldsymbol{v},$$

where $\boldsymbol{v} = A^{\mathrm{T}}\boldsymbol{e}_j$. Therefore, by expression (3), Algorithm 1 can be used to estimate the bounds on any entries of the inverse of the matrix A with $f(\lambda) = 1/\lambda$, and $\boldsymbol{u} = e_i$ and $\boldsymbol{v} = A^{\mathrm{T}}e_j$.

3.2.2. *Bounds on* $\mathrm{tr}(A^{-1})$

In order to bound $\mathrm{tr}(A^{-1})$, observe that

$$\mathrm{tr}(A^{-1}) = \tfrac{1}{2}\,\mathrm{tr}(A^{-1} + A^{-\mathrm{T}}) = \tfrac{1}{2}\,\mathrm{tr}((A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} + A(A^{\mathrm{T}}A)^{-1})$$

and

$$\begin{aligned} e_i^{\mathrm{T}}((A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} + A(A^{\mathrm{T}}A)^{-1})e_i &= e_i^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}e_i + e_i^{\mathrm{T}}A(A^{\mathrm{T}}A)^{-1}e_i \\ &= e_i^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}\boldsymbol{v} + \boldsymbol{v}^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}e_i \\ &= 2e_i^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}\boldsymbol{v}, \end{aligned}$$

where $\boldsymbol{v} = A^{\mathrm{T}}e_i$. From these relations, we see that Algorithm 1 can be used to estimate bounds on $e_i^{\mathrm{T}}(A^{\mathrm{T}}A)^{-1}\boldsymbol{v}$ with expression (3). Furthermore, since $(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}} + A(A^{\mathrm{T}}A)^{-1}$ is symmetric, we can apply Monte Carlo approach in Section 4, which requires the matrix to be symmetric.

3.2.3. *Bounds on* $\det(A)$

Let Eq. (6) apply to the matrix $A^{\mathrm{T}}A$ where we assume A is nonsingular, then we have

$$\det(A^{\mathrm{T}}A) = \exp(\mathrm{tr}(\ln(A^{\mathrm{T}}A))).$$

On the other hand, by the property of determinant, we have

$$\det(A^{\mathrm{T}}A) = \det(A^{\mathrm{T}})\det(A) = (\det(A))^2.$$

Therefore, let $f(\lambda) = \ln\lambda$, we can estimate $\mathrm{tr}(f(A^{\mathrm{T}}A))$ using the approach described in Section 3, and then

$$\det(A) = \pm\sqrt{\exp(\mathrm{tr}(\ln(A^{\mathrm{T}}A)))}.$$

The drawback with this approach is that the sign of the determinant is lost. But in the QCD application, we know the sign of the determinant in advance from the physical properties of the matrix involved [16, 20].

4. Monte Carlo approach and confidence interval

In this section, we will develop a Monte Carlo approach for bounding the quantity $\mathrm{tr}(f(A))$. Instead of applying Algorithms 1 n times for each diagonal element $f(A)_{ii}$, the Monte Carlo approach only applies Algorithm 1 m times to obtain an unbiased estimator of $\mathrm{tr}(f(A))$, where $m \ll n$. The saving in computational costs is significant. Probabilistic confidence bounds for the unbiased estimator are also presented in this section.

4.1. *Theory*

The Monte Carlo approach is based on the following proposition [11, 2].

Proposition 4.1. *Let H be an $n \times n$ symmetric matrix with $\operatorname{tr}(H) \neq 0$. Let V be the discrete random variable which takes the values 1 and -1 each with probability 0.5 and let z be a vector of n independent samples from V. Then $z^{\mathrm{T}}Hz$ is an unbiased estimator of $\operatorname{tr}(H)$, i.e.,*

$$E(z^{\mathrm{T}}Hz) = \operatorname{tr}(H),$$

and

$$\operatorname{var}(z^{\mathrm{T}}Hz) = 2 \sum_{i \neq j} h_{ij}^2.$$

As shown in Section 3, the problems of estimating the bounds of $\operatorname{tr}(A^{-1})$ and $\det(A)$ are reduced to bounding the trace of a symmetric matrix H, which is a function of A. Specifically, when A is symmetric positive definite, $H = A^{-1}$ or $H = \ln A$. When A is not symmetric positive definite, $H = A^{-1} + A^{-\mathrm{T}}$ and $H = \ln(A^{\mathrm{T}}A)$. Therefore, we can use Proposition 4.1 to obtain an unbiased estimator of $\operatorname{tr}(H)$.

In practice, we take m sample vectors z_i as described in Proposition 4.1, and then use Algorithm 1 (see Section 2) to obtain a lower bound L_i and an upper bound U_i of the quantity $z_i^{\mathrm{T}}Hz_i$,

$$L_i \leqslant z_i^{\mathrm{T}}Hz_i \leqslant U_i, \tag{9}$$

and furthermore, we have $E(z_i^{\mathrm{T}}Hz_i) = \operatorname{tr}(H)$, for $i = 1, 2, \ldots, m$. By taking the mean of the m computed lower and upper bounds L_i and U_i, we have

$$\frac{1}{m}\sum_{i=1}^{m} L_i \leqslant \frac{1}{m}\sum_{i=1}^{m} z_i^{\mathrm{T}}Hz_i \leqslant \frac{1}{m}\sum_{i=1}^{m} U_i. \tag{10}$$

It is natural to expect that with a suitable sample size m, the mean of the computed bounds yields a good estimation of the quantity $\operatorname{tr}(H)$.

To quantitatively assess the quality of such estimation, we now turn to the question of confidence bounds of the estimation. In other words, we want to find an interval so that the exact value of $\operatorname{tr}(H)$ is in such interval with probability p, where $0 < p < 1$. There is a Hoeffding's exponential inequality in probability theory which can be used to derive such confidence bounds [17].

Proposition 4.2 (Hoeffding's inequality). *Let $w_1, w_2, \ldots, w_m$ be independent random variables with zero means and bounded ranges $a_i \leqslant w_i \leqslant b_i$. Then for each $\eta > 0$,*

$$P(w_1 + w_2 + \cdots + w_m \geqslant \eta) \leqslant \exp\left(\frac{-2\eta^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right),$$

and

$$P(|w_1 + w_2 + \cdots + w_m| \geqslant \eta) \leqslant 2\exp\left(\frac{-2\eta^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right).$$

To apply Hoeffding's inequality, we let $w_i = z_i^{\mathrm{T}}Hz_i - \operatorname{tr}(H)$. Since z_i are taken as independent random vectors, w_i are independent random variables. From Proposition 4.1, w_i has zero means (i.e. $E(w_i) = 0$). Furthermore, from (9), we also know that w_i has bounded ranges

$$L_{\min} - \operatorname{tr}(H) \leqslant w_i \leqslant U_{\max} - \operatorname{tr}(H)$$

for all i, $1 \leqslant i \leqslant m$, where $U_{\max} = \max\{U_i\}$ and $L_{\min} = \min\{L_i\}$.

Hence, by Hoeffding's inequality, we have the following probabilistic bounds for the mean of m samples $z_i^T H z_i$,

$$P\left(\left|\frac{1}{m}\sum_{i=1}^{m} z_i^T H z_i - \mathrm{tr}(H)\right| \geqslant \frac{\eta}{m}\right) \leqslant 2\exp\left(\frac{-2\eta^2}{d}\right), \tag{11}$$

where $d = m(U_{\max} - L_{\min})^2$ and $\eta > 0$ is a tolerance value, which is related to the probability on the right-hand side of the inequality. In other words, inequality (11) tells us that

$$P\left(\frac{1}{m}\sum_{i=1}^{m} z_i^T H z_i - \frac{\eta}{m} < \mathrm{tr}(H) < \frac{1}{m}\sum_{i=1}^{m} z_i^T H z_i + \frac{\eta}{m}\right) > 1 - 2\exp\left(\frac{-2\eta^2}{d}\right).$$

Then from the bounds (10), we have

$$P\left(\frac{1}{m}\sum_{i=1}^{m} L_i - \frac{\eta}{m} < \mathrm{tr}(H) < \frac{1}{m}\sum_{i=1}^{m} U_i + \frac{\eta}{m}\right) > 1 - 2\exp\left(\frac{-2\eta^2}{d}\right). \tag{12}$$

Therefore, we conclude that the trace of H is in the interval

$$\left(\frac{1}{m}\sum_{i=1}^{m} L_i - \frac{\eta}{m}, \frac{1}{m}\sum_{i=1}^{m} U_i + \frac{\eta}{m}\right),$$

with probability $1 - 2\exp(-2\eta^2/d)$.

If we specify the probability p in (12), i.e. $p = 1 - 2\exp(-2\eta^2/d)$, then solving this equality for η/m, yields

$$\frac{\eta}{m} = \sqrt{-\frac{1}{2m}(U_{\max} - L_{\min})^2 \ln\left(\frac{1-p}{2}\right)}. \tag{13}$$

Since $(U_{\max} - L_{\min})^2$ is bounded by $2n^2 \| H \|^2$, we see that with a fixed value of p, $\eta/m \to 0$ as $m \to \infty$, i.e., the confidence interval is essentially given by the means of the computed bounds.

4.2. Algorithm

We now present a Monte Carlo algorithm which computes an unbiased estimator of $\mathrm{tr}(f(A))$ where A is symmetric positive definite. The algorithm will also return a confidence interval with user specified probability. Algorithms for the nonsymmetric positive-definite case can be developed in a similar framework using the strategies discussed in Section 3.

First, we note that, heuristically, the means of L_i and U_i are very sharp bounds of $z_i^T A z_i$, which is an unbiased estimator of $\mathrm{tr}(H)$. It would be ideal if we could have a sharp confidence interval, i.e., η/m is small. However, from Eq. (13), we may have to choose quite a large number of samples m. It would be too expensive to run such a large number of samples. Instead, we suggest to choose a fixed number of samples m and the probability p to compute the corresponding confidence interval. Here is the algorithm based on the Monte Carlo approach.

Algorithm 2. Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\lambda(A) \subset [a, b]$. Let m be a chosen number of samples. Then the following algorithm computes (a) an unbiased estimator I_p of the

quantity $\mathrm{tr}(f(A))$, and (b) a confidence interval (L_p, U_p) such that $\mathrm{tr}(f(A)) \in (L_p, U_p)$ with a user specified probability p, where $0 < p < 1$.

```
1. For $j = 1, 2, \cdots, m$
    1. Generate $n$-vector $z_j$ whose elements are uniformly distributed in the interval $(0,1)$.
    2. For $i = 1:n$, if $z_j(i) < 0.5$, then $z_j(i) = -1$, otherwise, $z_j(i) = 1$.
    3. Apply Algorithm 1 with Gauss–Radau rule to obtain a lower bound $L_j$ and an upper
       bound $U_j$ of the quantity $z_j^T f(A) z_j$
    4. $I(j) = \frac{1}{2j}\sum_{i=1}^{j}(L_i + U_i)$
    5. $L_{\min} = \min\{L_{\min}, L_j\}$
    6. $U_{\max} = \max\{U_{\max}, U_j\}$
    7. $\eta^2 = -0.5j(U_{\max} - L_{\min})^2(\ln(1-p)/2)$
    8. $L_p(j) = \frac{1}{j}\sum_{i=1}^{j} L_i - \frac{\eta}{j}$
    9. $U_p(j) = \frac{1}{j}\sum_{i=1}^{j} U_i + \frac{\eta}{j}$
2. End
3. $I_p = I(m)$
4. $L_p = L_p(m)$
5. $U_p = L_p(m)$
```

5. Numerical examples

All algorithms are implemented in Matlab 4.2 with sparse matrix computation functions. The tolerance value ε for the stopping criterion of Algorithm 1 is set to be 10^{-4}. The bounds a and b for eigenvalues of A are estimated by the Gerschgorin circle theorem. It is observed that the convergence behavior of the algorithm is less sensitive to the values a and b. The poor estimation of a and b usually only adds a few more iterations. The so-called "exact" values reported in this section are the quantities computed by using full dense or direct sparse matrix computation methods. For example, the trace of the inverse of a matrix is computed by first computing the inverse explicitly and then compute the trace. All numerical experiments are carried out on an SUN Sparc 10 workstations with IEEE double precision floating point operations.

5.1. Bounds on $(A^{-1})_{ii}$ and $(A^{-1})_{ij}$

Example 1 (*Linear heat flow matrix*). This test matrix is from [15]. The matrix is from the discretization of the linear heat flow problem using the simplest implicit finite difference method. The coefficient matrix A of the resulted linear system of equations is a $k^2 \times k^2$ block tridiagonal matrix

$$A = \begin{pmatrix} D & C & & & \\ C & D & C & & \\ & C & D & \ddots & \\ & & \ddots & \ddots & C \\ & & & C & D \end{pmatrix},$$

Table 1

Linear heat flow matrix, $n = 900$

$(A^{-1})_{ii}$		Gauss–Radau			
i	"Exact" value	Iter	Lower bound L_i	Upper bound U_i	$U_i - L_i$
1	$5.7020150e-01$	4	$5.7020115e-01$	$5.7020202e-01$	$8.68e-7$
22	$5.7792259e-01$	4	$5.7792195e-01$	$5.7792349e-01$	$1.54e-6$
32	$5.8626306e-01$	4	$5.8626209e-01$	$5.8626430e-01$	$2.21e-6$

where D is a $k \times k$ tridiagonal matrix

$$D = \begin{pmatrix} 1+4v & -v & & & \\ -v & 1+4v & -v & & \\ & -v & 1+4v & \ddots & \\ & & \ddots & \ddots & -v \\ & & & -v & 1+4v \end{pmatrix},$$

and $C = \mathrm{diag}(-v, -v, \dots, -v)$, $v = \Delta t/h^2$, Δt is timestep, h is spacing interval. It can be shown that A is symmetric positive definite for $v > 0$.

Table 1 shows the computed bounds for some diagonal elements of the matrix A with $n = 900$ by Algorithm 1. In fact, the diagonal elements of the inverse of A only have three different values corresponding to the 3, 4 or 5 nonzeros in the rows of the original matrix. In [18], formulations for the bounds of three different values of the diagonal elements of the inverse of A are given. Using their formulas, we have the bounds for the three representative values of the diagonal elements of the inverse of A, where $v = 0.2$ and $\mathrm{cond}(A) = 2.5853$.

$$5.676320033658228e-01 < (A^{-1})_{1,1} < 5.750190403655749e-01$$

$$5.730060434231142e-01 < (A^{-1})_{22,22} < 5.866283249460820e-01$$

$$5.786555786555786e-01 < (A^{-1})_{32,32} < 5.974842767295597e-01$$

It clearly shows that the bounds L_i and U_i are much sharper bounds. In addition, Algorithm 1 is a very efficient algorithm, it only takes 4 iterations to obtain convergent bounds.

For bounding the off-diagonal elements, we use formula (3) and Algorithm 1. Table 2 shows the computed bounds.

Example 2 (*Vicsek fractal Hamiltonian (VFH) matrix*). This test matrix is from the transverse vibration of a Vicsek fractal [13, 19, 12]. The fractal is self-similarly constructed. The first generator of the Vicsek gasket matrix H composed of five atoms with its four outermost atoms anchored to

Table 2

Linear heat flow matrix, $n = 900$

$(A^{-1})_{ij}$			Gauss–Radau			
i	j	"Exact" value	Iter	Lower bound L_i	Upper bound U_i	$U_i - L_i$
2	1	$6.5906786e-02$	4	$6.5906436e-02$	$6.5907171e-02$	$7.35e-07$
20	21	$6.6837061e-02$	4	$6.6836507e-02$	$6.6837584e-02$	$1.07e-06$
200	181	$1.6328011e-17$	4	$-1.4359500e-06$	$1.4359500e-06$	$2.87e-06$
200	700	$3.9509029e-19$	4	$-1.7276332e-06$	$1.7276332e-06$	$3.45e-06$
899	895	$1.1189572e-04$	4	$1.1106335e-04$	$1.1273010e-04$	$1.66e-06$

a rigid boundary site, i.e., the first stage uses atoms as its cell H_1, the kth stage H_k uses $(k-1)$th fractal as its cell:

$$H_k = \begin{bmatrix} H_{k-1} & V_{k-1}^{(1)^T} & V_{k-1}^{(2)^T} & V_{k-1}^{(3)^T} & V_{k-1}^{(4)^T} \\ V_{k-1}^{(1)} & H_{k-1} & 0 & 0 & 0 \\ V_{k-1}^{(2)} & 0 & H_{k-1} & 0 & 0 \\ V_{k-1}^{(3)} & 0 & 0 & H_{k-1} & 0 \\ V_{k-1}^{(4)} & 0 & 0 & 0 & H_{k-1} \end{bmatrix},$$

with

$$H_1 = \begin{bmatrix} -4 & 1 & 1 & 1 & 1 \\ 1 & -2 & 0 & 0 & 0 \\ 1 & 0 & -2 & 0 & 0 \\ 1 & 0 & 0 & -2 & 0 \\ 1 & 0 & 0 & 0 & -2 \end{bmatrix}.$$

The order n_k of the matrix H_k is $n_k = 5n_{k-1}$. $V_{k-1}^{(i)}$ are the interaction matrices linking one generation to the next generation. Let $p_{1,1} = 3$, $p_{2,1} = 2$, $p_{3,1} = 5$, $p_{4,1} = 4$ and

$$p_{1,k-1} = 5^{k-2}(p_{1,1} - 1) + p_{1,k-2}, \qquad p_{2,k-1} = 5^{k-2}(p_{2,1} - 1) + p_{2,k-2},$$

$$p_{3,k-1} = 5^{k-2}(p_{3,1} - 1) + p_{3,k-2}, \qquad p_{4,k-1} = 5^{k-2}(p_{4,1} - 1) + p_{4,k-2},$$

for $k = 3, 4, \cdots$. Then $V_{k-1}^{(1)}$ is a zero matrix except with one at $(p_{1,k-1}, p_{2,k-1})$ entry, $V_{k-1}^{(2)}$ is a zero matrix except with one at $(p_{2,k-1}, p_{1,k-1})$ entry, $V_{k-1}^{(3)}$ is a zero matrix except with one at $(p_{3,k-1}, p_{4,k-1})$ entry and $V_{k-1}^{(4)}$ is a zero matrix except with one at $(p_{4,k-1}, p_{3,k-1})$ entry. It is easy to see that H_k is symmetric positive definite.

One of the interesting computational problems related to H_k is to find some diagonal elements (and trace) of the inverse of the matrix H_k. Tables 3 and 4 show the estimated bounds for some diagonal elements of the matrix with dimension $n = 625$ and $n = 3125$. Again, we obtained sharp bounds in only 13 to 20 iterations.

Table 3

Vicsek fractal Hamiltonian matrix, $n = 625$				
$(A^{-1})_{ii}$	Gauss–Radau			
i	Iter	Lower bound L_i	Upper bound U_i	$U_i - L_i$
1	13	$9.480026e-01$	$9.480197e-01$	$1.71e-05$
100	15	$1.100520e+00$	$1.100527e+00$	$7.00e-06$
301	11	$9.242992e-01$	$9.243184e-01$	$2.62e-05$
625	13	$6.440017e-01$	$6.440054e-01$	$3.70e-06$
Vicsek fractal Hamiltonian matrix, $n = 3125$				
1	19	$9.4801416e-01$	$9.4801776e-01$	$3.60e-06$
100	20	$1.1005253e+00$	$1.1005302e+00$	$4.90e-06$
2000	19	$1.1003685e+00$	$1.1003786e+00$	$1.01e-05$
3125	16	$6.4400234e-01$	$6.4401100e-01$	$8.66e-06$

Table 4

Poisson matrix, $n = 900$, $\text{cond}(A) = 564.92$					
$(A^{-1})_{ij}$		Gauss–Radau			
i	j	Iter	Lower bound L_i	Upper bound U_i	$U_i - L_i$
2	1	16,37;11,29	$1.0456440e-01$	$1.0484172e-01$	$2.7732e-04$
1	900	13,30;13,32	$-5.7004377e-05$	$1.5611762e-04$	$2.1312e-04$
10	90	26,50;22,46	$1.5135698e-04$	$4.4211369e-04$	$2.9076e-04$
41	42	36,54;13,31	$2.2938427e-01$	$2.2965834e-01$	$2.7407e-04$
58	59	24,50;14,35	$1.9190948e-01$	$1.9226997e-01$	$3.6050e-04$
450	449	35,51;26,49	$1.7901387e-01$	$1.7926657e-01$	$2.5269e-04$
550	750	38,53;39,54	$3.9886981e-03$	$4.1810605e-03$	$1.9236e-04$
600	602	33,52;24,52	$8.2673761e-04$	$1.0917413e-03$	$2.6500e-04$
650	750	40,54;39,53	$1.6301914e-02$	$1.6479777e-02$	$1.7786e-04$

Example 3 (*Poisson matrix*). In this example, the matrix of order k^2 is a block tridiagonal matrix from the 5-point central difference discretization of the 2-D Poisson's equation on a $k \times k$ square mesh. Table 4 shows the results for bounding some off-diagonal elements of the matrix using expression (3) and Algorithm 1 with the Gauss–Radau rule.

Note that in the column "iter" there are four numbers in each row. The first two represent the number of iterations in the Gauss–Radau rule for the lower and upper bounds of $y^T f(A) y$, where $y = e_i + e_j$. The last two are the number of iterations for the lower and upper bounds of $z^T f(A) z$, where $z = e_i - e_j$.

5.2. *Bounds on* $\text{tr}(A^{-1})$ *and* $\det(A)$

We now present numerical examples of Algorithm 2 for the Monte Carlo estimation of the quantities $\text{tr}(A^{-1})$ and $\det(A)$. Tables 5 and 6 are the summary of the results for different test

Table 5 Summary of the Monte Carlo approach for estimating $\mathrm{tr}(A^{-1})$

Matrix	n	"Exact"	Iter	Estimated	Rel. err	Confidence bounds
Poisson	900	$5.126e+02$	$30-50$	$5.020e+02$	2.0%	$(4.28e+02, 5.75e+02)$
VFH	625	$5.383e+02$	$12-21$	$5.366e+02$	0.3%	$(5.05e+02, 5.67e+02)$
Wathen	481	$2.681e+01$	$33-58$	$2.667e+01$	0.5%	$(2.58e+01, 2.75e+01)$
Lehmer	200	$2.000e+04$	$38-70$	$2.017e+04$	0.8%	$(1.86e+04, 2.16e+04)$

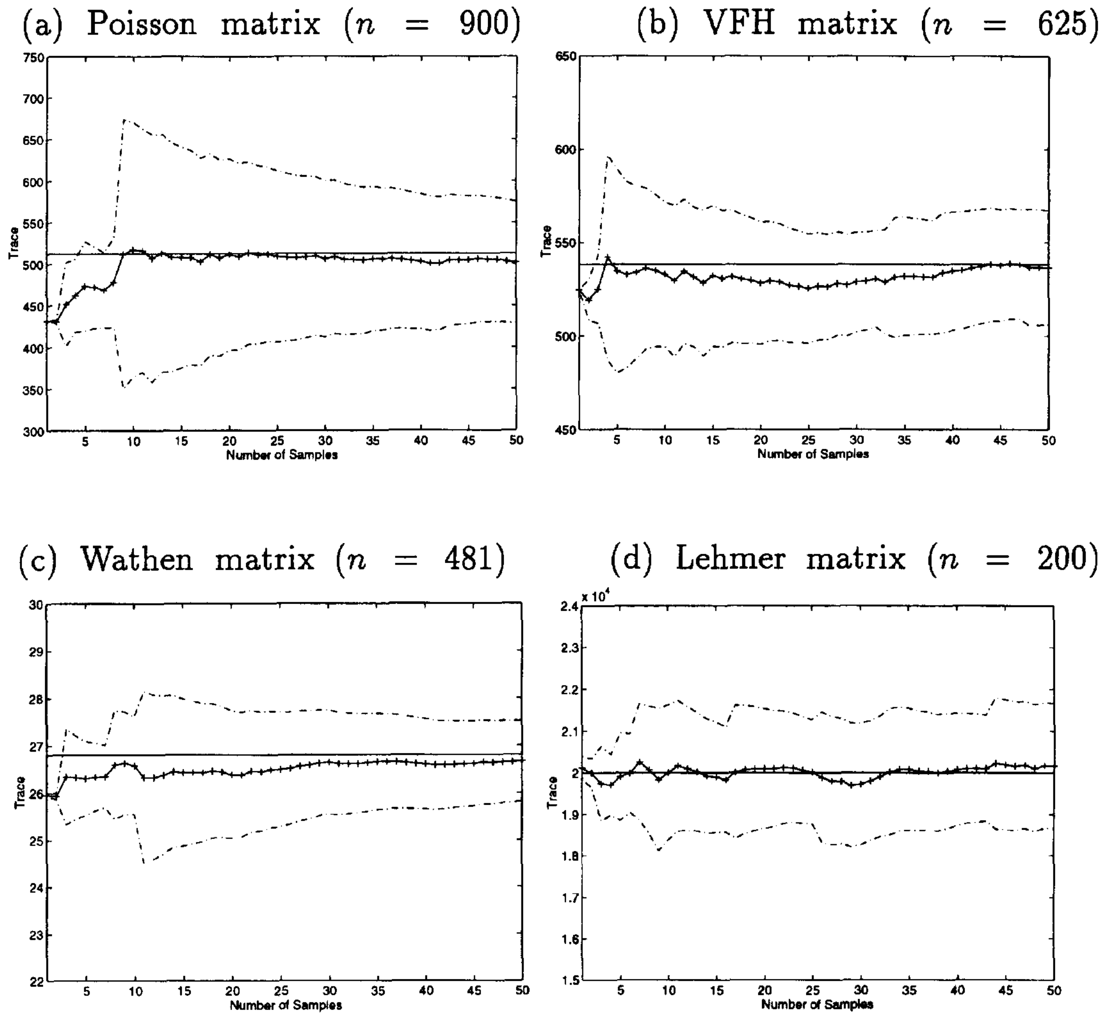

Fig. 1. Numerical results of the Monte Carlo approach (Algorithm 2) for estimating $\mathrm{tr}(A^{-1})$. The solid horizontal line is the exact value of $\mathrm{tr}(A^{-1})$. The solid line with $+$ is the estimated value. The dash-dot lines are the confidence intervals of the estimated values with probability 0.95.

Table 6 Summary of the Monte Carlo approach for estimating tr(ln A)

Matrix	n	"Exact"	Iter	Estimated	Rel. err	Confidence bounds
Poisson	900	$1.065e+03$	11 – 29	$1.060e+03$	0.4%	$(1.03e+03, 1.08e+03)$
VFH	625	$3.677e+02$	10 – 14	$3.661e+02$	0.4%	$(3.38e+02, 3.94e+02)$
H flow	900	$5.643e+01$	4	$5.669e+01$	0.4%	(5.42e + ι.,5.92e + 01)
Pei	300	$5.707e+00$	2 – 3	$5.240e+00$	8.2%	$(-1.83e+0, 1.23e+1)$

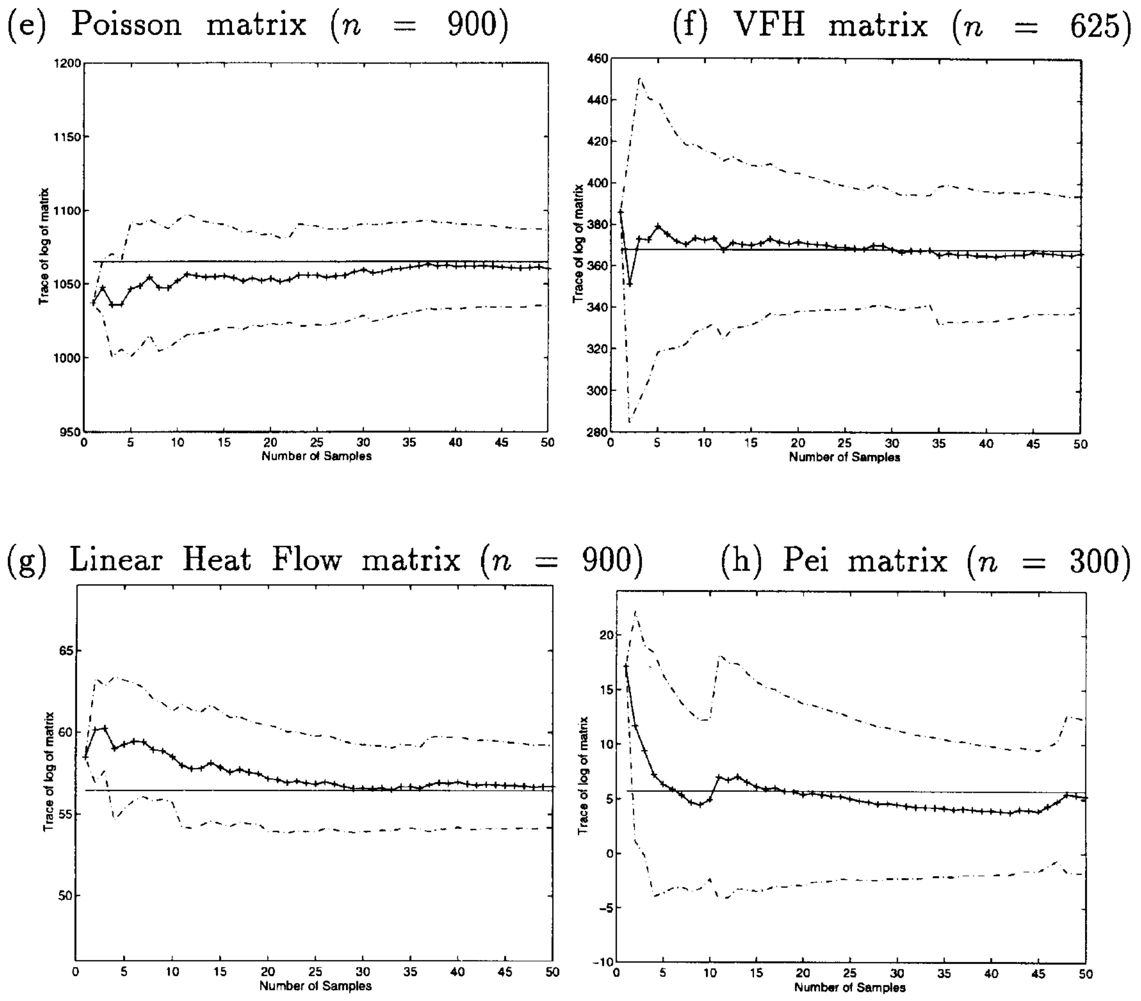

Fig. 2. Numerical results of the Monte Carlo approach (Algorithm 2) for estimating tr(ln(A)). The solid horizontal line is the exact value of tr(ln(A)). The solid line with + is the estimated value. The dash-dot lines are the confidence intervals of the estimated values with probability 0.95. Note that det(A) = exp(tr(ln(A))).

matrices. Figs. 1 and 2 show the history of the Monte Carlo approach where the number of sample vectors z_i is $m = 50$. Linear heat flow, VFH and Poisson test matrices have been described above. The additional three test matrices are from the Higham's test matrix collection [9]:

Wathen matrix: It is a consistent mass matrix for a regular $n_x \times n_y$ grid of 8-node (serendipity) elements in 2 space dimensions. The order of the matrix is $n = 3n_xn_y + 2n_x + 2n_y + 1$. In our numerical run, we let $n_x = n_y = 12$, then $n = 481$ and $\mathrm{cond}(A) = 829.19$.

Lehmer matrix: It is an $n \times n$ symmetric positive-definite matrix with entries $a_{ij} = i/j$ for $j \geqslant i$. The inverse of A is tridiagonal, and explicit formulas are known for its entries. $n \leqslant \mathrm{cond}(A) \leqslant 4n^2$. We tested the matrix with $n = 200$ and $\mathrm{cond}(A) = 4.8401e + 04$.

Pei matrix: $A = \alpha I + \mathbf{1}\mathbf{1}^{\mathrm{T}}$, where $\mathbf{1}$ is an n-vector with all ones. The matrix is singular for $\alpha = 0, -n$. We tested the matrix with $n = 300$, $\alpha = 1$ and $\mathrm{cond}(A) = 301$.

6. Conclusions and future work

An elegant approach for estimating the bounds of the quantity $\boldsymbol{u}^{\mathrm{T}}f(A)\boldsymbol{v}$ has been laid out earlier in [6, 7]. The theory of matrix moments, quadrature rules, orthogonal polynomials and the underlying Lanczos procedure are beautifully connected and turned into an efficient algorithm. In this paper, we have further developed the approach in a number of practical aspects. Something old, something new, and something borrowed. Preliminary numerical results for different matrices demonstrate the high-performance of the new approach.

Our near future work includes studying the ill-conditioning problems and preconditioning techniques, improving the confidence bounds, and investigating orthogonal polynomials with variable sign [21, 5] and related quadrature rules to solve the estimation problem with $\boldsymbol{u} \neq \boldsymbol{v}$ directly. The ultimate goal of this study is to develop truly efficient algorithms to solve such types of matrix computation problems where the matrices involved may be complex, non-Hermitian and of order millions. These matrix computation problems are arising in modern lattice QCD and other scientific computing fields.

Acknowledgements

The authors would like to thank Keh-Fei Liu and Shao-Jian Dong for bringing to our attention the applications of problem (1) in the lattice QCD, and thank Shao-Jian Dong for providing the VFH test matrix. ZB and MF would also like to thank Mai Zhou for introducing them to Hoeffding's inequality used in Section 4, and to Arnold Neumaier, who reminded us of the polarization expression (3).

References

[1] P. Davis and P. Rabinowitz, *Methods of Numerical Integration* (Academic Press, New York, 1984).
[2] S. Dong and K. Liu, Stochastic estimation with z_2 noise, *Phys. Lett. B* **328** (1994) 130–136.
[3] W. Gautschi, A survey of Gauss–Christoffel quadrature formulae, in: P.L. Bultzer and F. Feher, Eds., *E.B. Christoffel – the Influence of His Work on Mathematics and the Physical Sciences* (Birkhauser, Boston, 1981) 73–157.

[4] G. Golub, Some modified matrix eigenvalue problems, *SIAM Rev.* **15** (1973) 318–334.
[5] G. Golub and M. Gutknecht, Modified moments for indefinite weight functions, *Numer. Math.* **57** (1989) 607–624.
[6] G. Golub and G. Meurant, Matrics, moments and quadrature, Report SCCM-93-07, Computer Science Department, Stanford University, September 1993.
[7] G. Golub and Z. Strakos, Estimates in quadratic formulas, Report SCCM-93-08, Computer Science Department, Stanford University, September 1993.
[8] G. Golub and C. Van Loan, *Matrix Computations* (Johns Hopkins University Press, Baltimore, MD, 2nd edn., 1989).
[9] N.J. Higham, The test matrix toolbox for Matlab, Numerical Analysis Report No. 237, University of Manchester, England, Dec. 1993.
[10] G.M. Hockney, Comparison of inversion algorithms for Wilson fermions, *Nuclear Phys. B (Proc. Suppl.)* **17** (1990) 301–304.
[11] M. Hutchinson, A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines, *Commun. Statist. Simula.* **18** (1989) 1059–1076.
[12] C.S. Jayanthi, S.Y. Wu and J. Cocks, The nature of vibrational modes of a self-similar fractal, Technical report, Department of Physics, University of Louisville, KY, 1994.
[13] C. Kittel, *Introduction to Solid State Physics* (Wiley, New York, 1972).
[14] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Natl. Bur. Stand.* **45** (1950) 225–280.
[15] A.R. Mitchell and D.F. Griffiths, *The Finite Difference Method in Partial Differential Equations* (Wiley, New York, 1980).
[16] I. Montvay and G. Münster, *Quantum Fields on a Lattice* (Cambridge Univ. Press, Cambridge 1994).
[17] D. Pollard, *Convergence of Stochastic Processes* (Springer, New York, 1984).
[18] P. Robinson and J. Wathen, Variational bounds on the entries of the inverse of a matrix, *IMA J. Numer. Anal.* **12** (1992) 463–486.
[19] B. Sapoval, Th. Gobron and A. Margolina, Vibrations of fractal drums, *Phy. Rev. Lett.* **67** (1991) 2974–2977.
[20] J.C. Sexton and D.H. Weingarten, The numerical estimation of the error induced by the valence approximation, *Nuclear Phys. B* (*Proc. Suppl.*) **42** (1995) 361–363.
[21] G. Struble, Orthogonal polynomials: variable-signed weight functions, *Numer. Math.* **5** (1963) 88–94.
[22] S.Y. Wu, J.A. Cocks and C.S. Jayanthi, An accelerated inversion algorithm using the resolvent matrix method, *Comput. Phys. Commun.* **71** (1992) 125–133.